# CZ4041: Machine Learning

A comparative evaluation of different object detection models

Ong Kah Han (K2020282C)
Lim Haw Keat (K2020204B)
Pamela Sin Hui (K1920110A)

## Report Structure

1. Introduction
2. Problem Statement
3. Common Object Detection Datasets
4. Literature Review
    a. Faster R-CNN
    b. Single Shot Detector
    c. YOLO
5. Metric
6. Method
    a. Dataset & Data Pre-processing
7. Training Model
    a. Faster R-CNN
    b. Single Shot Detector
    c. YOLO
8. Result
9. Discussion
10. Reference List

# 1. Introduction

Computer vision is one of the key areas in the application of artificial intelligence. It involves the training of the computer to understand the images and videos with the aim of automating tasks (e.g. recognition), in the long run, that the human visual system can do. One of the hottest applications in computer vision is autonomous driving vehicles where computer vision is applied to detect obstacles, other vehicles and pedestrians, a critical and necessary safety feature.

Research on computer vision started as early as 1950 where it involved using basic neural networks to detect edges of objects and also to perform simple object classification. During the early 1950s to 70s, computer vision gained a lot of research funding support from government and private companies due to optimism about the potential future application of computer vision. However, the researchers underestimated the complexity of computer vision challenges as the computing resources available then could not support the systems to solve the real problems due to its high demand for computing power. As the hardware computing power could not keep up with the technical requirements, the research in computer vision slowed down until the 1980s - AI winter. [1] Computer vision then regained a lot of traction in 2010 due to a much greater availability of computer power, access to vast troves of data on the Internet and also break-through in deep learning techniques (e.g. convolutional neural networks in the 1990s).

Some common genres of computer vision tasks are edge detection, feature extraction, image classification, object detection and image segmentation. In recent years, a particular genre, object detection, made tremendous improvement in computer vision due to break-throughs in techniques using deep neural networks. Object detection, when given an input image, requires the model to not only classify objects found in the image but also to pinpoint their location by marking with bounding boxes. With object detection models reaching good accuracy, they can now be deployed, and are indeed widely used, in many real life applications such as pose estimation, self-driving vehicles, facial detection, security systems, industrial automation and medical imaging.

## 2. Problem Statement

Since the introduction and wide-spread adoption of deep learning techniques in 2012, it helped to speed up development of highly accurate object detection algorithms with transfer learning techniques. In this project, multiple object detection algorithms are studied to compare their performance. Faster R-CNN, SSD and YOLO algorithms are selected for comparison study. Based on the literature review so far, the initial thought is that single look detectors will consume more training time as compared to region-based detectors but single look detectors will predict results much faster during actual application.

## 3. Common Object Detection Datasets

There are object detection datasets available that provide images and their corresponding annotations for training, validating and testing object detection models. Below, we briefly describe these datasets and their size.

ImageNet dataset targets to generalize machine learning methods through large scale image databases that can be used for training and testing. ImageNet has around 10,000 classes with an average of 1000 images for each class.

PASCAL VOC challenge ran from 2005 up to 2012 focusing on object detection techniques, it has 20 classes in the dataset, with 11530 images for training and testing. The classes are person, bird, car, cow, dog, horse, sheep, airplane, bicycle, boat, bus, motorbike, train, bottle, chair, dining table, potted plant, sofa and television.

Common object in Context (COCO) dataset has 200,00 images with more than 500,000 object annotations in 80 classes.

The Open Images Dataset (OID) consists of approximately 9 million images annotated with image level labels, object bounding boxes, object segmentation masks, visual relationships and localized narratives. The 9 million images consists of 600 classes with 16 million bounding box annotations.

# 4. Literature review

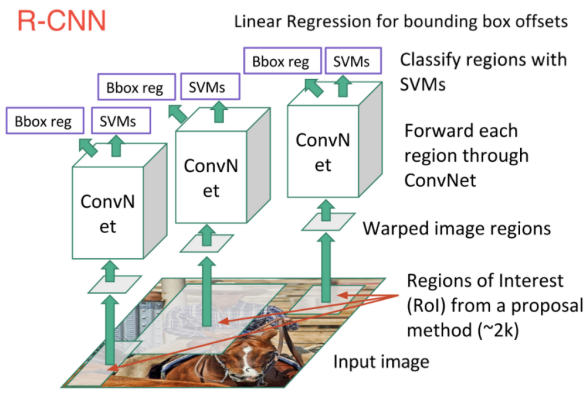
Object detection refers to recognizing and classifying every object in an image, while also locating each one of them by drawing a bounding box on it. Object detection is widely used in face detection, vehicle detection, security system, smart video surveillance and autonomous vehicles. There are many object detection algorithms such as Single Shot Multibox Detector (SSD), Region-based fully convolutional networks (R-FCN), Region-based Convolutional Neural Networks (R-CNN), Fast R-CNN, Faster R-CNN and You Only Look Once (YOLO).

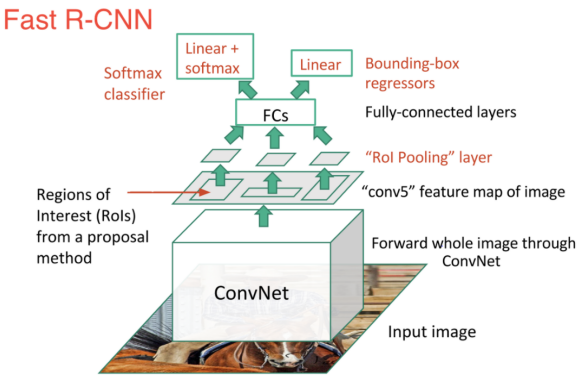
## Faster R-CNN

Faster R-CNN [3] is the evolution from R-CNN & Fast R-CNN. Regions with CNN features (R-CNN) published in 2014 where it uses selective search to propose possible regions of interest to generate 2000 region proposals and classify the region proposals using a standard CNN. The output of CNN is then fed into SVM for classification and linear regression for predicting and adjusting four values that can be used to draw the bounding box. The key challenges faced by R-CNN are heavy computation resources required to train the CNN as you need to classify 2000 region proposals per image.

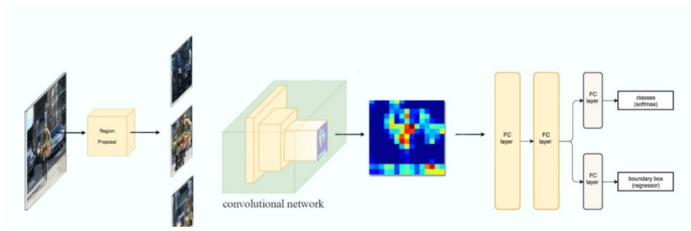
In early 2015, some of the drawbacks in R-CNN were resolved by introduction of Fast R-CNN. Fast R-CNN improved its detection speed by performing feature extraction prior to region proposal, where the input image is fed into the CNN to generate a convolution feature map, which means one feature map is fed in CNN for convolution operation instead of 2000 images. From the convolution feature map, we identify the region of proposals and feed it into a fully connected layer, which then performs image classification tasks using softmax and adjusting the bounding box location using linear regressors. In general, Fast R-CNN outperforms the R-CNN model but region proposal using selective search for Fast R-CNN is the potential area for improvement.

Faster R-CNN replaced the selective search process with a region proposal network (RPN). The input images are imputed into a pre-trained model which generates convolutional feature maps, the convolutional feature maps are then fed into a separate network which is called region proposal network to predict the region proposals. The predicted region proposals are then reshaped using a ROI pooling layer. The ROI pooling layer is used to classify the image within the proposed region and predict offset values for the bounding boxes.

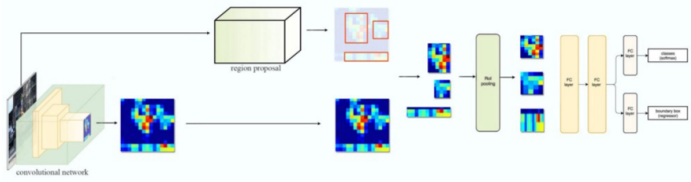



R-CNN

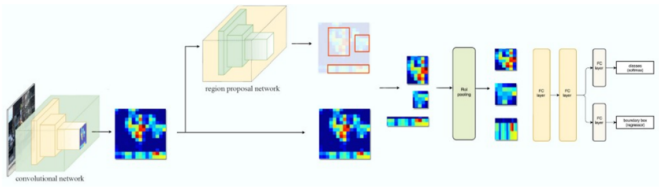



Fast R-CNN



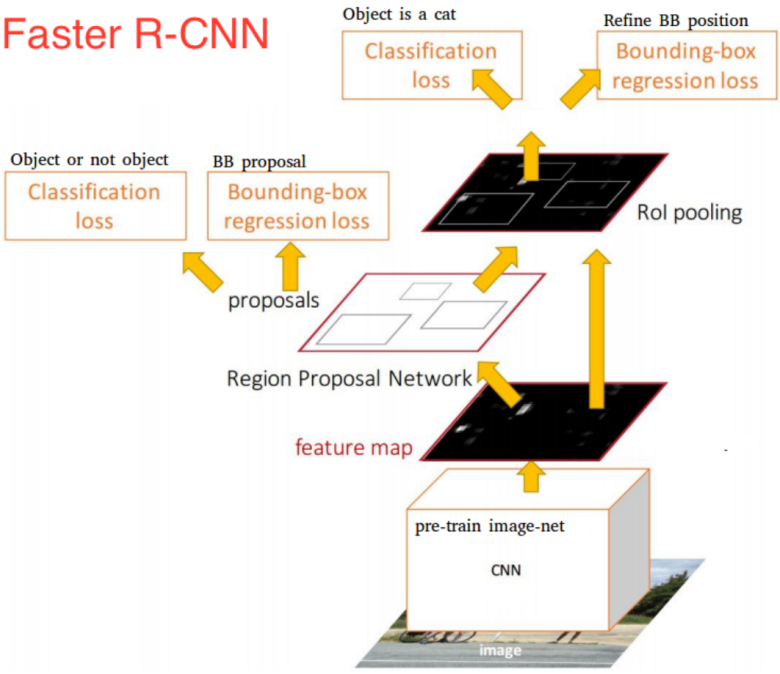

Apply region proposal on feature maps and form fixed size patches using ROI pooling.

Faster R-CNN




The external region proposal is replaced by an internal deep network.

https://jonathan-hui.medium.com/what-do-we-learn-from-region-based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377a7c9




Faster R-CNN contains two networks. The first network is a Region Proposal Network (RPN) used to propose regions, while the second is the detector network. RPN is a network used to propose multiple regions. RPN outputs a series of bounding boxes proposal that will feed into a classifier and regressor. The output of the classifier is determined either foreground or background, where background is defined as an area where the object is not present. For the area with IOU more than 0.5, it will be labeled as foreground class. The regressor is used to finetune the size of the bounding box. The bounding box with foreground class labelled in the form of feature map will be imputed into region of interest (ROI) pooling layer.

Anchor boxes are key features in Faster R-CNN where it replaced selective search algorithms in Fast R-CNN. The position, shape and size of anchor boxes are carefully selected to allow various types and sizes of objects to be detected.

Anchor centers throught the original image. Source: https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

The Faster R-CNN is jointly trained with 4 losses, which are calculated during the training. The losses are :

1. RPN classification - loss_rpn_cls (if the object is present or not)
2. RPN regression - loss_rpn_regr (Region of interest)
3. Detector Network classification - loss_class_cls
4. Detector Network classification - loss_class_regr

## SSD

In 2015, Wei Liu et al. published a paper outlining a single-look detector model which they named Single Shot Detector (SSD). [10] Similar to other single-look detectors like YOLO, instead of having a region proposal stage, SSD uses bounding boxes to detect objects based on a feature map produced by a backbone feature extractor.

In the 2015 paper, the authors used a reduced VGG-16 feature extractor (i.e. without the classification layer) as the backbone feature extractor. After that, based on the feature map, SSD uses a series of downsampling convolution layers to predict the positions and offsets of probable bounding boxes. Each layer's predicted bounding boxes are fed directly into the general detection layer. One particular useful outcome of the downsampling portion is that bounding box predictions are done on feature maps with different scales, thus allowing predictions to be done on objects of different sizes in the input image.

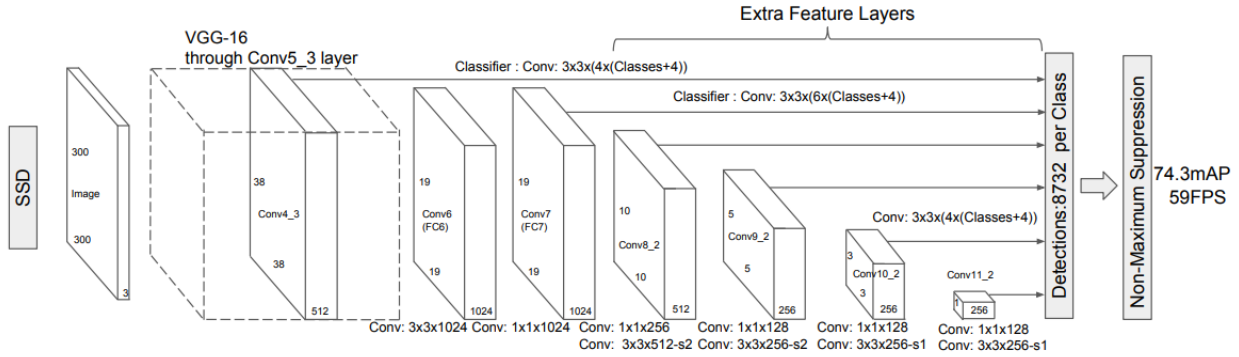



Figure extracted from [10]

However, unlike YOLO, SSD uses bounding boxes with pre-determined aspect ratios instead of producing them through k-means clustering. [11] In addition, SSD predicts bounding boxes without determining whether there is an object in a particular bounding box. In this manner, SSD requires the use of an additional "background" class to account for bounding boxes detecting objects in the background. These bounding box predictions for the "background" are then sieved out in the final predictions.

Also, as there may be many overlapping predicted bounding boxes, non-max suppression is used at the end to sieve out such overlapping predictions as well as predictions below a confidence threshold, leaving only the predicted bounding boxes with the highest confidence behind in the output.

Non-max suppression first discards all boxes with probability less than the confidence threshold, which is to discard bounding boxes assigned with a low probability. Secondly, while there are any remaining bounding boxes, pick the box with the largest probability, and output that as a prediction. Thirdly, any remaining box with the pre-determined IoU of say 0.5 with the output prediction is discarded. The second and third steps are repeated while there is still any

remaining boxes that have not been processed, until you have taken each of the boxes and either output it as a prediction, or discarded it as having too high IoU with one of the boxes you have output as your predicted position for one of the detected objects. If there are n classes of objects in our dataset, then non-max suppression is carried out n times, one on each of the output classes.

## YOLO - YOLOv1

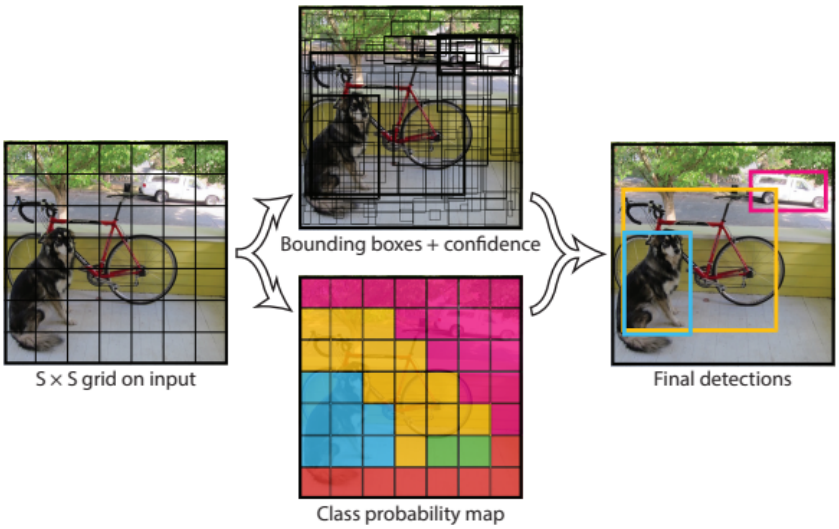
In the 2016 paper ”You Only Look Once: Unified, Real-Time Object Detection” by Joseph Redmon , Santosh Divvala, Ross Girshick and Ali Farhadi [19], YOLO was presented as a single neural network that predicts bounding boxes and class probabilities directly from full images in one evaluation as a single pipeline (single regression problem) that achieved fast results. A single convolutional network predicts multiple bounding boxes and class probabilities for those boxes by firstly resizing the image, running the convolutional network, and applying non-max suppression.

Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. The paper makes mention, for example, that Fast R-CNN, mistakes background patches in an image for objects because it is unable to see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

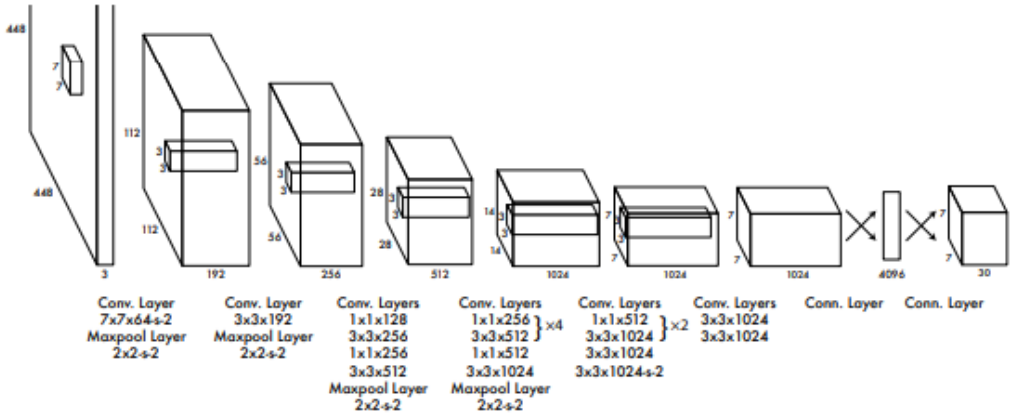
YOLO divides the input image into an S × S grid [17]. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. The confidence scores reflect how confident the model is that the box contains an object as well as how accurate it thinks the box it predicts is. If no object exists in that cell, the confidence scores should be zero, otherwise, the intersection over union (IOU) between the predicted box and the ground truth. Each grid cell also predicts C conditional class probabilities conditioned on the grid cell containing an object. Only one set of class probabilities per grid cell is predicted, regardless of the number of boxes B. At test time the conditional class probabilities and the individual box confidence predictions are multiplied together. This gives the class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

Sum-squared error is optimized, with the loss from bounding box coordinate predictions being increased and the loss from confidence predictions for boxes that do not contain objects being decreased through the use of hyperparameters in a multi-part loss function. This gives localization error higher weights than classification error to align better with the goal of maximizing average precision, and further addresses the fact that many grid cells do not contain an object.

The 2016 paper states limitations of YOLO, including the loss function treating errors the same in small bounding boxes versus large bounding boxes with the latter having a much greater effect on IOU, further stating that the main source of error is incorrect localizations.




For an image of S × S grid, each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. The predictions are S × S × (B ∗ 5 + C) tensors




The YOLO detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. The final layer predicts class probabilities and box coordinates.

## YOLO - YOLOv4

Since the original 2016 paper, “YOLO9000: Better, Faster, Stronger” [18] was published in late 2016 and “YOLOv3: An Incremental Improvement ” was published in 2018 by Joseph Redmon and Ali Farhad. In 2020, “YOLOv4: Optimal Speed and Accuracy of Object Detection” by Alexey Bochkovskiy, Chien-Yao Wang and Hong-Yuan Mark Liao was published [16]. As one of the latest implementations of YOLO, the YOLOv4 architecture is further explored for the purposes of this project.

YOLOv4 consists of a backbone, neck and head. Further, the 2020 paper documents methods to optimise performance under “Bag of Freebies” or “Bag of Specials” [14] [15]. Bag of Freebies are methods that can make the object detector receive better accuracy without increasing the inference cost. These methods only change the training strategy or only increase the training cost. Whereast Bag of Specials are plugin modules and post-processing methods that only increase the inference cost by a small amount but can improve the accuracy of object detection

## YOLO - YOLOv4 (Backbone)

For the backbone feature extractor, YOLOv4 utilizes Cross-Stage-Partial-connections (CSP connections) with Darknet-53.

Darknet-53 is a DenseNet formed by composing multiple Dense Blocks with a transition layer in between made up of convolution and pooling. A Dense Block contains multiple convolution layers with each layer composed of batch normalization, ReLU, and followed by convolution. Instead of using the output of the last layer only, each layer takes the output of all previous layers as well as the original as its input.

CSP separates the input feature maps of the DenseBlock into two parts. The first part bypasses the DenseBlock and becomes part of the input to the next transition layer. The second part will go through the Dense block. The new design reduces the computational complexity by separating the input into two parts — with only one going through the Dense Block.

For the backbone, one example of a Bag of Freebies deployed is CutMix and Mosaic data augmentation. In CutMix, a portion of an image is cut-and-paste over another image. The ground truth labels are readjusted proportionally to the area of the patches.

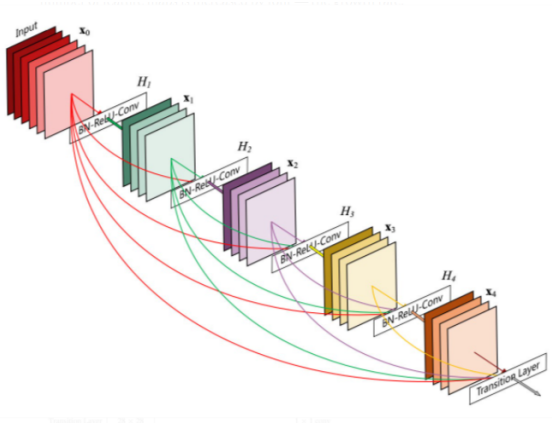



Illustration of a Dense Block

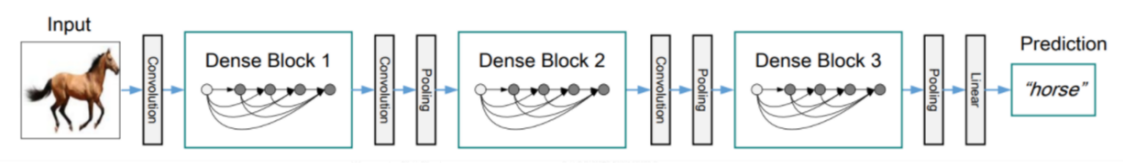



Illustration of a DenseNet

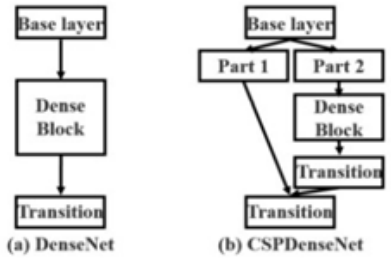



Illustration of DenseNet versus CSPDenseNet

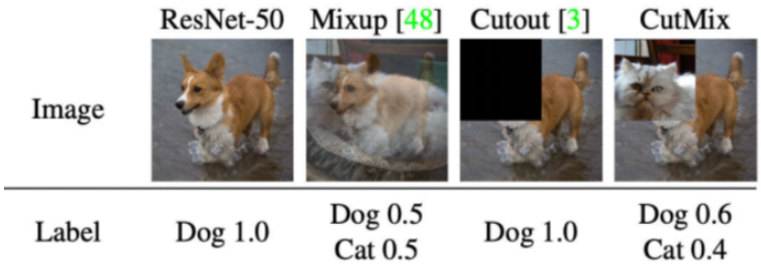



Illustration of CutMix

## YOLO - YOLOv4 (Neck)

For the neck, a hierarchy structure is produced with the head probing feature maps at different spatial resolutions. To enrich the information that feeds into the head, neighbouring feature maps coming from the bottom-up stream and the top-down stream are added together element-wise or concatenated before feeding into the head. Therefore, the head's input will contain spatial-rich information from the bottom-up stream and the semantic rich information from the top-down stream. In YOLOv4, the neck is composed of the Spatial Pyramid Pooling (SPP) additional module and Path Aggregation Network (PAN) [20].

SPP is used to acquire both fine and coarse information by simultaneously pooling on multiple kernel sizes (1,5,9,13).

PAN is a technique that leverages information in layers close to the input by conveying features from different backbone levels to the head. A bottom-up path is augmented to make low-layer information easier to propagate to the top. In FPN, the localized spatial information traveled upward (red arrow in the below diagram). PAN introduced a short-cut path (the green path) which only takes about 10 layers to go to the top $N_5$ layer, making fine-grain localized information available to top layers. In YOLOv4, feature maps are concatenated (instead of added) together. PAN reduces duplicated predictions and utilises information from multiple feature maps by fusing the information together from all layers first using element-wise max operation.

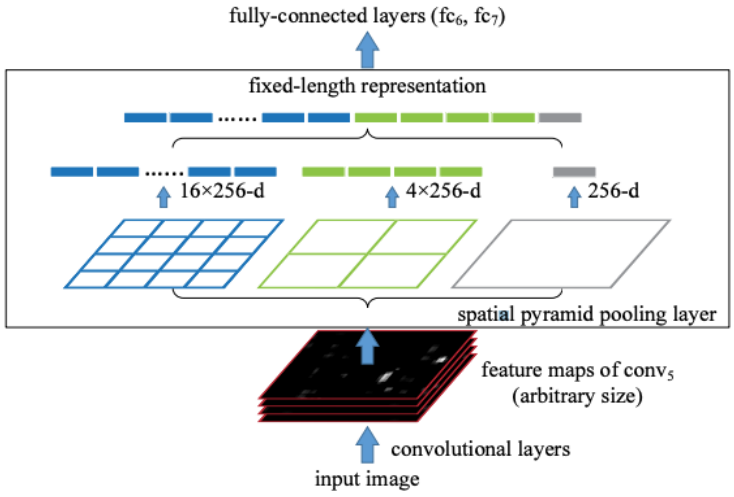
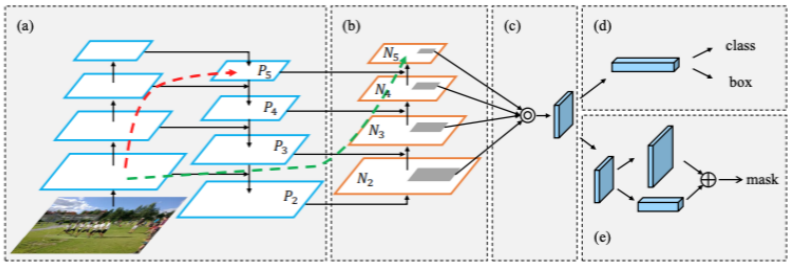


Illustration of SSP [21]




(a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. The channel dimension of feature maps in (a) and (b) for brevity [22]

## YOLO - YOLOv4 (Head)

YOLOv4 uses YOLOv3 as the head for object detection.

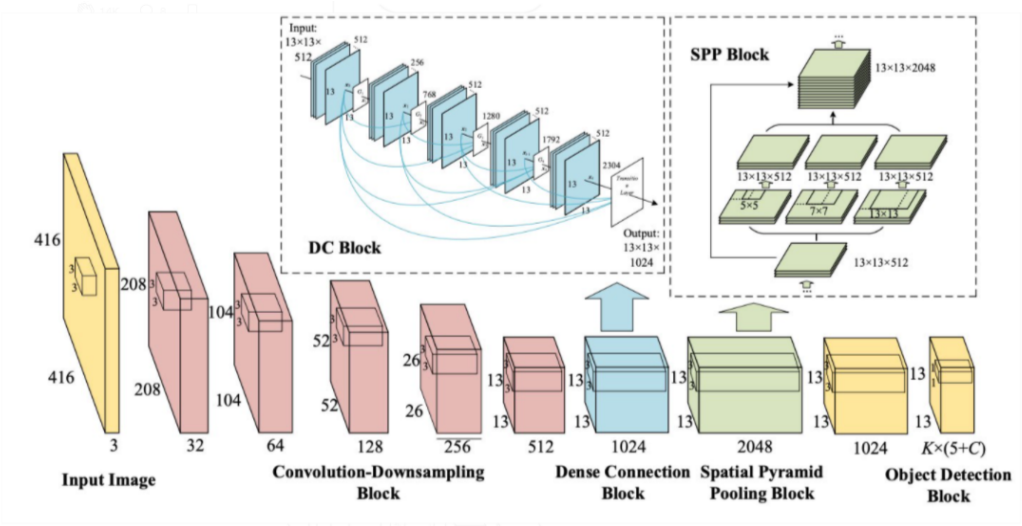



Illustration of YOLOv4

# 5. Metric [6]

For supervised learning, there are various metrics to measure performance of classification and regressors tasks. Metrics for classification and regressors are different due to their different nature. Metrics used in classification tasks are accuracy, precision and/or recall.

Intersection over union (IoU) and Mean Average Precision (mAP) are common metrics used in object detection where IoU focuses on evaluating localization tasks, and mAP focuses on evaluating detection tasks.

Intersection over union (IoU) refers to the ratio of overlapping area of ground truth (Bgt) and predicted area (Bp) to the total area. The IoU is then used in conjunction with algorithms to train localization tasks.

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

Mean average precision (mAP) is the product of precision and recall of the detected bounding boxes, which range from 0 to 100. Higher number of mAP indicates better detection algorithms.

Real time object detection is another key factor to ensure ability to run the object detection model in the real time application devices. Hence test time (second per image) is monitored to compare the performance for each model.

# 6. Method

## Dataset & Data pre-processing

The project dataset contains images picked from the Open Images Dataset (OID), which contains a total of 9 million images over 600 classes with a total of 16 million bounding box annotations. The overview description of OIDV6 states it is the “largest existing dataset with object location annotations”. It states that for the training set, 90% of the boxes were manually drawn by professional annotators at Google using the efficient extreme clicking interface, and the remaining 10% is produced semi-automatically (where these are human verified to have IoU>0.7) using an enhanced version of the method. For the training set, bounding boxes for the human body parts and “Mammal” class were limited to a certain number of images due to a large number of instances. Further, a single box around groups of objects was drawn if there were more than 5 instances which were heavily occluding each other and physically touching. For the validation and test sets, an exhaustive box annotation for all object instances was provided and all boxes were manually drawn. [13]

For the project, images with 4 classes were selected - Man, Woman, Boy and Girl. These 4 classes do not exist in the image datasets which the pre-trained models were trained on (e.g. ILVSRC, VOC, COCO etc) were picked. Image datasets like VOC subsume them into an overarching Human class, hence requiring the models to further classify human images into the sub-classes will be a good test of their abilities. The constraints of Google Drive storage space, Colabatory run-out times limiting GPU usage and combined training times were factors in determining the 10,000 images limit for the project. Further, given the relatively “small” number of images decided, a split of 80%, 10% and 10% for the train, validation and test sets respectively was determined for sufficient images for the training phase.

The full training set annotations ('oidv6-train-annotations-bbox.csv') and class description codes ('class-descriptions-boxable.csv') were downloaded from the OID platform. The core information of ImageID, LabelName, XMin, XMax, YMin, YMax (X and Y start and end of the ground truth bounding boxes in the normalized minimum and maximum of X and Y value of real coordinates of the original image) for our specific 4 object classes was then filtered. This OID training set for the 4 object classes was determined to be highly skewed in favour of the Man class, in which 67.67% of the 558,692 images have at least one Man object in them, and 57.42% of the objects are in the Man class. On the opposite side of the spectrum, the OID training set for the 4 object classes had an underrepresentation of the Boy class, where a mere 8.22% of the images have at least one Boy object in them, and 3.54% of the objects are in the Boy class.

To achieve a more uniform distribution of classes for the project training set so that the object detection models are trained approximately equally on all classes, the selection of a certain percentage of the underrepresented Girl and Boy classes was curated. The count of the number of Boy and Girl objects per image was taken. For the first stage, firstly, the most underrepresented Boy class was addressed by sorting the ImageID dataframe in descending order of the count of Boy objects, and taking an empirically derived top 2,500 images. As such,

from the start, a fixed 2,500 images that have at least one Boy object inside is added to the training set, and these are the top 2,500 images in terms of the Boy object count, raising the Boy object representation in regards to the total number of objects in the training set. For the first stage, secondly, the same method is used to address the next most underrepresented Girl class by taking the empirically derived top 1,000 images when sorting the ImageID dataframe in descending order of the count of Girl objects. For the second stage, to obtain the remaining images, a random selection of the empirically derived 1,000 images containing at least one Boy object, 1,200 images containing at least one Girl object, 1,600 images containing at least one Woman object and 700 images containing at least one Man object was taken.

The empirical derivation of the numbers of images taken into the training set at each turn was done over a few iterations manually, where at each iteration the selected project train set was evaluated for the representation of each class in the training set. The numbers of images in the first stage and second stage are akin to knobs, where increasing the values in the first stage raises the base representation of the Boy and Girl classes, and the values in the second stage randomly select images from the OID training set based on the criterion of there being at least a single object for the selected class in question. The selection of images at each iteration as well as the evaluation of each selected project train set was completed instantaneously upon running the code, thus the final tuned number of images to add into the project train set for the first and second stage could be quickly derived. The annotations for the selected 8,000 train images was saved in a xlsx file, and the 8,000 ImageIDs saved in a single txt file in a “train/$IMAGE_ID” format per row. As specified in OIDv6, a provided downloader.py file and dependencies were installed to download the 8,000 images specified in the txt file.

The project validation set and project test set was obtained through random sampling of the OID validation set and OID test set respectively. The assumption is that the OID validation and OID test set distributions are reflective of the “real world” dataset that is likely to contain forms of class imbalance. It is noted that the OID validation set and OID test set have similar distributions of data.

| OID training set | | |
|---|---|---|
| **Object class** | **No. of images that have at least one object class inside** | **No. of objects** |
| Man | 378,077 (67.67%)* | 1,418,594 (57.42%)** |
| Woman | 265,635 (47.55%) | 767,337 (31.06%) |
| Boy | 45,938 (8.22%) | 87,555 (3.54%) |
| Girl | 93,731 (16.78%) | 197,155 (7.98%) |
| Man, Woman, Boy, Girl | There are 558,692 images containing Man, Woman, Boy, Girl objects | 2,470,641 |

*Calculation: 378,077 / 558,692 = 67.67%
**Calculation: 1,418,594 / 2,470,641 = 57.42%

| Project training set | | |
|---|---|---|
| **Object class** | **No. of images that have at least one object class inside** | **No. of objects** |
| Man | 4,061 (50.76%) | 22,765 (24.33%) |
| Woman | 4,153 (51.91%) | 20,856 (22.29%) |
| Boy | 3,811 (47.64%) | 24,438 (26.12%) |
| Girl | 3,722 (46.53%) | 25,509 (27.26%) |
| Man, Woman, Boy, Girl | There are 8,000 images containing Man, Woman, Boy, Girl objects | 93,568 |

| OID validation set | | |
|---|---|---|
| **Object class** | **No. of images that have at least one object class inside** | **No. of objects** |
| Man | 3,734 (58.93%) | 8,493 (47.24%) |
| Woman | 2,775 (43.80%) | 4,767 (26.52%) |
| Boy | 579 (9.14%) | 1,095 (6.09%) |
| Girl | 2,135 (33.70%) | 3,622 (20.15%) |
| Man, Woman, Boy, Girl | There are 6,336 images containing Man, Woman, Boy, Girl objects | 17,977 |

| Project validation set | | |
|---|---|---|
| **Object class** | **No. of images that have at least one object class inside** | **No. of objects** |
| Man | 580 (58.00%) | 1,228 (43.76%) |
| Woman | 442 (44.20%) | 815 (29.04%) |
| Boy | 92 (9.20%) | 158 (5.63%) |
| Girl | 355 (35.50%) | 605 (21.56%) |
| Man, Woman, Boy, Girl | There are 1,000 images containing Man, Woman, Boy, Girl objects | 2,806 |

| OID test set | | |
|---|---|---|
| **Object class** | **No. of images that have at least one object class inside** | **No. of objects** |
| Man | 11,350 (59.44%) | 26,231 (47.42%) |
| Woman | 8,259 (43.25%) | 14,551 (26.31%) |
| Boy | 1,874 (9.81%) | 3,493 (26.31%) |
| Girl | 6,515 (34.12%) | 11,040 (19.96%) |
| Man, Woman, Boy, Girl | There are 19,096 images containing Man, Woman, Boy, Girl objects | 55,315 |

| Project test set | | |
|---|---|---|
| **Object class** | **No. of images that have at least one object class inside** | **No. of objects** |
| Man | 594 (59.40%) | 1,338 (45.81%) |
| Woman | 433 (43.30%) | 775 (26.53%) |
| Boy | 97 (9.70%) | 192 (6.57%) |
| Girl | 356 (35.60%) | 616 (21.09%) |
| Man, Woman, Boy, Girl | There are 1,000 images containing Man, Woman, Boy, Girl objects | 2,921 |

# 7. Training models

To maintain consistency, the Faster R-CNN, SSD and YOLO models were trained on the same “Project training set” of 8,000 images defined above. The dataset images were also resized to 416 x 416 pixel images, whilst preserving the aspect ratio. A batch size of 64 images was selected. Colabatory, a platform for users to execute python code through the browser, was selected as the platform as it provides free GPUs (notwithstanding fluctuating usage limits).

## Faster R-CNN

In this exercise, The Faster R-CNN is implemented in Keras/Tensor 1.x platform by Ying Han, Xu [5]. Ying Han implemented the Faster R-CNN model in the paper by Shao Qiao [7] together with a pretrained model VGG-16.

2 phases of training are conducted. For the initial pre-training phase, 6000 images were randomly selected from the dataset. The image is then resized to 300 pixels. The training of models was completed within 12 hours. In order to perform a good comparison between different models, all models are retrained with the same dataset as described above as input with more control on training parameters as well. For example, the image size is set at 416 pixels, IoU is set to 0.5, the learning rate is set at 0.001. The selected train/test dataset is captured in an excel file. Code is written to convert the format of the dataset excel file into relevant format. For the model implemented by Ying Han, it requires the dataset to be organized in the sequence of file path, location of bounding box and class. Code is modified to convert the dataset file into the format that is fitted into the Faster R-CNN model. It was observed that choice of dataset does influence the performance of model training. The mAP drop from 40% to 32%

Resnet pre-trained model was implemented as an alternative during training phases as well, where it was found that the mAP is 0.503, which is almost doubled as compared to VGG pretrained model. However, it is observed that the distribution of AP among classes is not realistic and required further attention. Observations showed that AP for girl class is close to 0.9 accuracy, which will be listed as a recommendation to look into details in future works.

## Single Shot Detection (SSD)

In this exercise, a Single Shot Detection (SSD) model implemented using a Keras/Tensorflow 1.x platform was used. [12] This SSD model implementation hews closely to that in the 2015 paper by Wei Liu et al. in that a reduced VGG-16 feature extractor (i.e. without the classification layer) forms the base of the SSD model preloaded with weights from pre-training on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

Two phases of training for the SSD model were conducted. For the initial pre-training phase, 6000 images were extracted for training through random sampling. They are then downsized (while preserving aspect ratio) to no larger than 512 x 512 pixels. A model was then trained for

250 epochs with batch size set at 32 and the learning rate set at 0.001 for the first 200 epochs and thereafter changed to 0.0001 for the last 50 epochs.

For the training phase, a separate new SSD model was trained on the actual training and validation sets. These training and validation sets were resized (while preserving aspect ratio) to no larger than 416 x 416 pixels. To speed up the training process, transfer learning was employed by initialising this separate new SSD model with the trained model from the initial pre-training phase (with the best validation loss metric) to start the training.

Even though the training phase uses training and validation sets of different sizes with a more equal distribution across the different classes compared to initial pre-training phase, it is opined that this should not be a material concern as, firstly, the issue of different sizes does not materially affect the pretrained model's ability to extract features as it is a matter of scaling which can be easily handled by the model. Secondly, a more balanced distribution of class labels will not materially affect the pre-trained model's operations as the resulting classification layer remains, architecturally, the same and the more balanced distribution of class labels serves to refine, not overhaul, the model's training.

One small point to note is that because this implementation was originally used for the PASCAL VOC set, the input annotations were expected to be in the PASCAL VOC XML format. As such, a further step to pre-process the data was required to transform the annotations into PASCAL VOC XML files in order for the training, validation and testing to be conducted smoothly.

## YOLO

In this exercise, a You Only Look Once (YOLO) model implemented using a DarkNet platform at [GitHub - AlexeyAB/darknet: yolov4] was used. The GitHub repository is stated in the 2020 paper "YOLOv4: Optimal Speed and Accuracy of Object Detection".

Darknet was cloned from AlexeyAB's GitHub repository, settings were adjusted to enable OPENCV and GPU for darknet, and darknet was built. To take advantage of transfer learning, pre-trained YOLOv4 weights that was trained on the COCO dataset (with 80 classes) was downloaded. The pre-trained weights for the convolutional layers are downloaded to reduce training times in order for the model to converge more rapidly.

The annotation files were converted into the YOLOv4 format, <object-class> <x> <y> <width> <height>, where <x> <y> <width> <height> are float values relative to width and height of image (0.0 to 1.0], and <x> <y> is the center of the bounding box.

The configuration file was edited to specify hyperparameters such as height and width of images (416 x 416), batch size (64), maximum numbers of batches to train on (16,000, attempting the double recommended of (No. of classes) * 2000 = 8,000), learning rate of 0.001, number of batches before decreasing the learning rate (12,800 and 14,400, which is the recommended (80% of max_batches), (90% of max_batches)), number of filters in each

convolutional layer before YOLO layers (the recommended (No. of classes + 5) * 3 = 27 was used) and number of classes (4).

A few other setup steps were done such as firstly, creating an obj.data file to specify the Google Drive path of txt files that contains the paths of the train, validation and test images. Secondly, running scripts to obtain the relative paths of the train, validation and test images.

During the training process, the last weights are being saved at regular intervals. Separately the best weights that produce the best mAP on the validation set are saved.

Command line flags such as the threshold (-thresh) flag was used to add a threshold for confidences on detections.

The code for the above implementation can be downloaded from the following link:

Faster R-CNN:
https://github.com/shermon-ong/faster-r-cnn

SSD:
https://github.com/shermon-ong/SSD

YOLO:
https://github.com/shermon-ong/yolov4

# 8. Result

The summary of comparison of results across the three models is shown below:

| Average Precision (AP) for confidence threshold of 0.50 | | | |
|---|---|---|---|
| **Class** | **Faster R-CNN(VGG)** | **SSD** | **Yolo(CSPDarknet53)** |
| Man | 55.38% | 31.25% | 52.57% |
| Woman | 38.65% | 27.59% | 47.45% |
| Boy | 09.70% | 20.39% | 31.04% |
| Girl | 54.38% | 11.36% | 42.42% |
| **mAP (%)** | 32.9% | 22.65% | 43.37% |
| **Test time (sec/Img)** | 0.4170 | 0.0115 | 0.0332 |

Generally speaking, in terms of test timings, SSD is the fastest, followed by YOLO before Faster R-CNN. In terms of mAP across all classes, SSD has the lowest mAP and YOLO was able to outperform Faster R-CNN on the mAP level as well as across all classes.

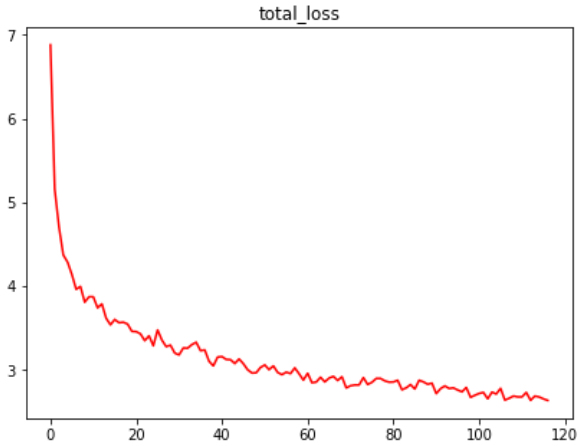
When looking at the AP across classes on an intra-model basis, it is noted that the Man class has consistently higher AP as compared to the other classes while the Boy class performed the worst in 2 of the 3 models (SSD has the Girl class performing the worst).

The models are more adept at detecting the Man class. Conversely, the Boy class does poorly. In doing ground-truth labels for the Google Open Images annotations for Woman and Girl are overlapping.

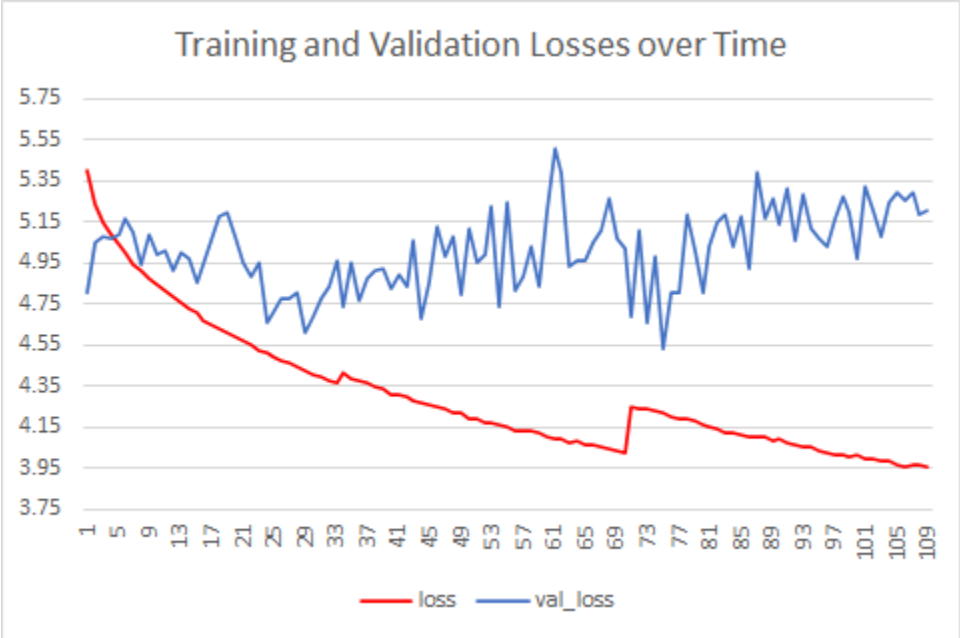
Below are losses from the Faster R-CNN training process. 4 losses are calculated during the training.

## Faster R-CNN

Training loss over epochs

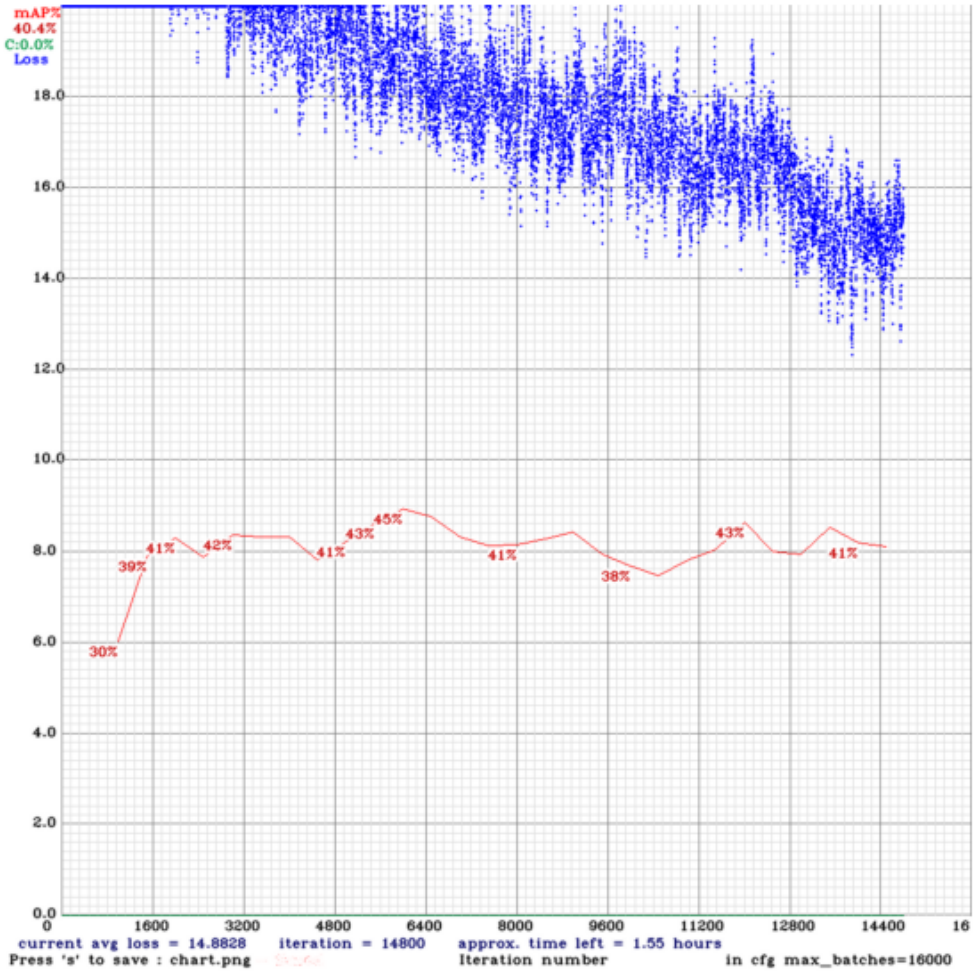



## SSD

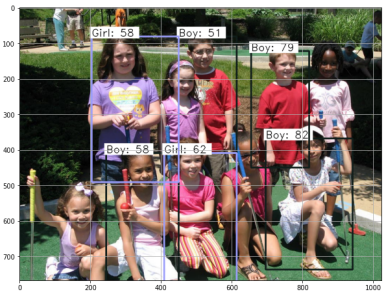
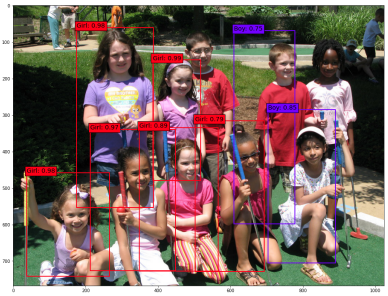
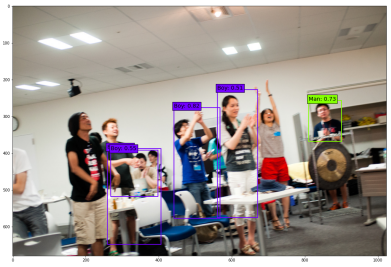


## YOLO

Training loss and validation set mAP over number of batches



Below are some randomly selected results to show predictions of each model:

| Faster R-CNN | YOLOv4 | SSD |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# 9. Discussion

## Loss

As trite as it might be, it must be noted that training loss decreases over time as the model undergoes more epochs of training. However, the same cannot be said for validation loss as it can fluctuate in a less-than-predictable manner. As such, even though the general principle is that a model should stop training when overfitting occurs, knowing when such overfitting has occurred seems to be more of an art than science. This also results in the models being trained over different numbers of epochs as overfitting (and hence cessation of training) occurs at different epochs for different models.

## Training and test time

All three models were run using Colab and a similar method was used to estimate training time and test times for each model.

For training times, it was observed that Faster R-CNN took around 500 seconds per epoch, YOLO took around 500 second per epoch, SSD took around 2000 seconds per epoch. For test timings (see Results section), the models performed as expected in that SSD was the fastest, followed by YOLO before Faster R-CNN.

This is in line with the general observation that single-look detectors generate predictions more quickly than region-based detectors. For training times, it is also aligned with our expectation that region-based detectors will consume lesser computation time as compared to single look detectors.

## Performance

In terms of mAP across all classes, SSD has the lowest mAP as expected. More unexpectedly, YOLO was able to outperform Faster R-CNN on the mAP level as well as across all classes.

When looking at the AP across classes on an intra-model basis, it is noted that the Man class has consistently higher AP as compared to the other classes while the Boy class performed the worst in 2 of the 3 models (SSD has the Girl class performing the worst).

Comparing between model families (region-based vs single-look), we see that SSD and Faster R-CNN performed in line with expectations that single-look models will be faster but with lower accuracy while the converse is true for region-based models. However, it is interesting to note that YOLO, despite being a single-look model, was able to perform faster and more accurately compared to Faster R-CNN. As at the time of writing this report, it is unclear as to the reason behind this and future research into increasing the dataset size and/or tweaking other hyperparameters or refining data augmentation would help to shed more light on this.

It is possible that results were materially affected by the different backbones. The YOLOv4 architecture utilises a newer backbone, while Faster-RCNN and SSD used an older backbone.

## Limitations encountered

As this is a project looking at the relative performance of applied or implemented models, this project focuses more on implementation compared to the architecture/technical aspects. Given that the models were coded and implemented by another creator, it would not be accurate to say that sufficient in-depth knowledge of the models was possessed by us to further optimise or even “hack” them to wring the best performance out of the models.

Also, data augmentation and metric evaluation were embedded in the implementation of each model. As such, they may not be consistent across all 3 models’ implementations. The project had proceeded on the assumption that they produce similar outcomes without any material differences. However, it would always be desirable to have data augmentation and metric evaluation done in a consistent and similar way across all 3 models.

Another limitation is the relatively small dataset sizes used. This is because of various restrictions (e.g. insufficient storage space for files, insufficient GPU memory space etc) that necessitated the usage of small datasets. If given more resources, the models could be trained on a much larger training dataset which should, in most cases, lead to better training and hence possibly more accurate outcomes.

## Possible areas for future research

This research is not the be-all-end-all for any person interested in this area. One can consider the following possible areas for future research:

1. Testing the effect of ground truth bounding box size on the mAP (i.e. whether models are equally adept at detecting both small and large objects)
2. Mapping the 4 classes into 2 super-classes - Male and Female and see if the models are able to perform as, if not more, effectively
3. Using the full training set from Open Images to train the models
4. Varying the IoU and confidence thresholds to see their effects on the mAP

# 10. Reference list


[1]https://www.motionmetrics.com/how-artificial-intelligence-revolutionized-computer-vision-a-brief-history/#:~:text=Computer%20vision%20began%20in%20earnest,that%20could%20transform%20the%20world.

[2]https://www.pulsarplatform.com/blog/2018/brief-history-computer-vision-vertical-ai-image-recognition/

[3]https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/

[4]https://github.com/RockyXu66/Faster_RCNN_for_Open_Images_Dataset_Keras

[5]https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a

[6]Paolo Galeone, Hands-on Neural networks with Tensorflow 2.0,Packt, 2019

[7]Shaoqing Ren, et , Faster R-CNN: towards real-time object detection with region proposal networks. 2016

[8]https://towardsdatascience.com/understanding-fast-r-cnn-and-faster-r-cnn-for-object-detection-adbb55653d97

[9]https://jonathan-hui.medium.com/what-do-we-learn-from-region-based-object-detectors-faster-r-cnn-r-fcn-fpn-7e354377a7c9

[10]https://arxiv.org/pdf/1512.02325.pdf

[11]https://www.jeremyjordan.me/object-detection-one-stage/

[12]https://github.com/pierluigiferrari/ssd_keras

[13]https://storage.googleapis.com/openimages/web/factsfigures.html

[14]https://towardsdatascience.com/yolo-v4-optimal-speed-accuracy-for-object-detection-79896ed47b50

[15]https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

[16]https://arxiv.org/pdf/2004.10934.pdf

[17]https://www.youtube.com/watch?v=ZILIbUvp5lk

[18]https://arxiv.org/pdf/1612.08242.pdf

[19]https://arxiv.org/pdf/1506.02640.pdf

[20]https://medium.com/axinc-ai/yolov4-a-machine-learning-model-to-detect-the-position-and-type-of-an-object-4f108ed0507b

[21]https://arxiv.org/pdf/1406.4729.pdf

[22]https://arxiv.org/pdf/1803.01534.pdf